Computer Science • Vol. 9 • 2008

K. Korcyl, T. Szymocha* W. Funika, J. Kitowski, R. Slota**
K. Balos, L. Dutka, K. Guzy, T. Kryza, J. Pieczykolan***

THE ATLAS EXPERIMENT ON-LINE MONITORING AND FILTERING AS AN EXAMPLE OF REAL-TIME APPLICATION

The ATLAS detector, recording LHC particles' interactions, produces events with rate of 40 MHz and size of 1.6 MB. The processes with new and interesting physics phenomena are very rare, thus an efficient on-line filtering system (trigger) is necessary. The asynchronous part of that system relays on few thousands of computing nodes running the filtering software. Applying refined filtering criteria results in increase of processing times what may lead to lack of processing resources installed on CERN site. We propose extension to this part of the system based on submission of the real-time filtering tasks into the Grid.

Keywords: *high energy physics, real-time procesing, trigger system, remote farms*

SYSTEM MONITORINGU I FILTRACJI EKSPERYMENTU ATLAS JAKO PRZYKŁAD APLIKACJI CZASU RZECZYWISTEGO

Detektor ATLAS, rejestrujący zderzenia protonów rozpędzanych w zderzaczu LHC, będzie generował przypadki o rozmiarze 1.6 MB z częstotliwością 40 MHz. Aby wyselekcjonować bardzo rzadko występujące przypadki z interesującymi oddziaływaniami fizycznymi, konieczne będzie zastosowanie wydajnego systemu filtracji (trigger). Część asynchroniczna takiego systemu wykorzystuje kilka tysięcy komputerów, na których wykonywane jest oprogramowanie filtrujące. Zwiększenie selektywności systemu wymaga zwiększenia czasu procesowania, co może doprowadzić do wyczerpania zasobów komputerowych zainstalowanych w CERN-ie. Proponujemy rozszerzenie tej części systemu poprzez umożliwienie wykonywania oprogramowania filtrującego w czasie rzeczywistym na komputerach w środowisku gridowym.

Słowa kluczowe: *fizyka wysokich energii, przetwarzanie w czasie rzeczywistym, system filtracji, zdalne farmy*

* Institute of Nuclear Physics Polish Academy of Sciences, Krakow, Poland,
Krzysztof.Korcyl@ifj.edu.pl
** Institute of Computer Science, AGH University of Science and Technology, Krakow, Poland
*** ACC CYFRONET AGH, Krakow, Poland,

1. Introduction

The ATLAS detector [1], will be recording interactions of protons beams with particles grouped in bunches and accelerated by the LHC accelerator [2] to energy 7 TeV. The bunch crossing rate of 40 MHz together with an average 25 interactions per crossing yields 1 GHz interaction rate. The interactions with interesting physics phenomena are very rare in range of tens fraction of Hz, thus an efficient on-line filtering system (trigger), capable to reduce the rate by 10 orders of magnitude is necessary. Such system would guide the data acquisition system to pull out from the data stream only interesting interactions and send them to the permanent storage for further physics analysis. The average size of ATLAS event is 1.6 MB and expected final rate of 200 Hz (interesting interactions and samples of background processes). The ATLAS trigger system is composed of two parts. The synchronous part, using coarse channels granularity, is based on processing implemented in the programmable logic devices FPGAs hence it is characterized by a fixed latency and is tightly coupled with the LHC timing. The asynchronous part of the system uses fine channel granularity and is distributed over a few thousands of computing cores running the filtering software. This part is further subdivided into two stages. At the earlier stage, the algorithms, running for events accepted by the synchronous part, fetch data from a subset of detector's buffers where interesting phenomena has been registered. However, their extended processing time may lead to buffer blockade and introduction of dead time. The last stage of the trigger system uses full event images, assembled during the Event Buiding process from the all detector buffers and stored on local disk drives. The selectivity of the trigger depends on a number and quality of filtering criteria and the more refined criterion the more processing power it needs. Thus making the trigger more selective may result in increasing demand for processing power what may lead to lack of processing resources installed on CERN site. We propose extension to the system based on submission of the real-time filtering tasks into the Grid.

2. Delegating processing tasks to resources offered by Grid

The ATLAS experiment implemented the trigger and data acquisition systems with assumption that all computing power (ca. 5000 CPUs) will be available at CERN site. In case the TDAQ system would have to filter higher event rate or make the trigger more refined by using more time for the filtering algorithms, some additional CPUs would be needed and these could be allocated from the Grid. The current organization of the ATLAS TDAQ system together with proposed extension is presented in Figure 1. Computers running the Sub Farm Interface (SFI) tasks collect data from all detector's channels for a given interaction to form an event image (the Event Building). There are almost one hundred of the SFI computers performing in parallel these tasks. When an event is completed the space in the detector's buffers occupied by the event data can be freed. The event image can then be sent to the last stage of filtering, the Event Filter, carried out by the nodes running the Event

Filter Deamon (EFD) process. The EFD communicates with the SFI, stores the event data in the RAM internal buffer and passes the pointer to the Processing Tasks (PTs) executed in the same node. A number of PTs can run in parallel on multi-CPU boxes or multi-core CPUs. The PTs perform the filtering tasks and after completion return decision and some additional information to the EFD. In case the event is accepted, the EFD sends the event data with processing results to the Sub Farm Output (SFO) node which relays it on to the system with permanent storage. As it is likely that the PT may crash on analysis of the unknown data patterns originated from new physic phenomena, the RAM space with events is mapped onto a file to avoid loosing events (the event data will be sent to permanent storage for further analysis).

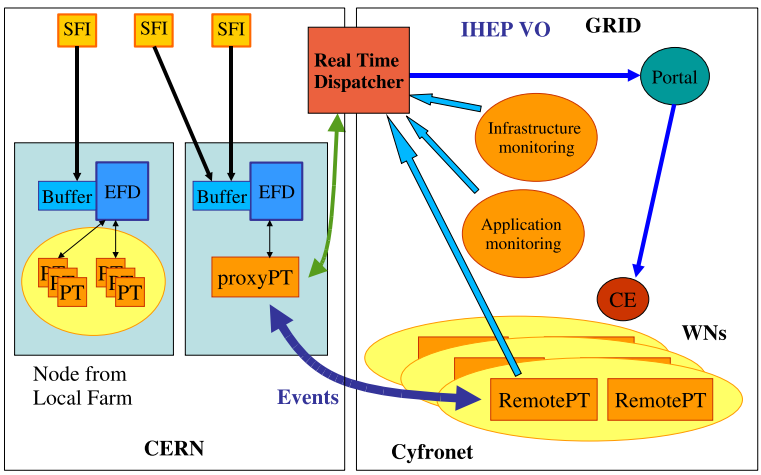

Fig. 1. Organization of the last stage of the current ATLAS TDAQ system with proposed extension. The leftmost box represents Event Filter local node with EFD and PT processes. The part of the figure to the right shows proposed extension where the PT is splitted into proxyPT running at CERN together with the EFD whereas the other part, the remotePT is located in Grid farm. The Real Time Dispatcher and IHEP Virtual Organization provide necessary infrastructure

The configuration of the ATLAS TDAQ system is entirely static as the configuration data is picked up from the database. All nodes collaborating in the system, from detector buffers to the SFO, are configured, initialized, monitored and controlled by the ATLAS TDAQ software.

To profit from the Grid remote resources with minimal modifications to the existing ATLAS TDAQ system we propose to break the processing task (PT) into the ProxyPT and RemotePT. The ProxyPT will be instantiated, monitored and controlled by the standard ATLAS TDAQ software in an ordinary way. When the event data will be received by the EFD and the pointer to it passed to the ProxyPT, the latter will communicate with RemotePT, pass on the event data and wait for results

of filtering work done at the remote site. When the RemotePT will finish processing the decision together with some additional results will be communicated back to the ProxyPT. The ProxyPT will pass on the data to the EFD. The EFD will continue operation as the processing would have been perfomed at the local machine. A large number of ProxyPT tasks can be instantiated on each of the EFD nodes as they do not consume local CPU resources for processing. As the ProxyPT will be paired with RemotePT for each event, the number of RemotePTs can be adjusted dynamically.

The infrastructure needed to support that idea together with schematic view of operation is presented in the right part of Figure 1. The ProxyPT will communicate with the Real Time Dispatcher (RTD) to find machine where the data will be sent to and which will run the RemotePT task. Only machines which have been certified can be used for that purpose. It is the task of the Interactive High Eenergy Physics Virtual Organization (IHEP VO) to certify, that the machine has been upgraded with the latest version of the analysis software together with the latest update to various databases used in the processing. The IHEP VO is also responsible for launching pilot jobs to the Grid to allocate machines where the RemotePT will subsequently be started to be ready to receive event data for processing.

Implementation details on the IHEP VO and the RTD will be presented in the following chapters.

3. Grid Real-Time Event Environment

3.1. IHEP VO Management

The IHEP VO is dedicated to support demanding HEP application. It is responsible for providing environment necessary to run the application using resources offered by the Grid. Available resources are carefully checked during certification procedure [3]. If all necessary conditions are fulfilled (like performance, stability, data privacy), SLA [4] with resources provider is signed and resources are certified and included in the IHEP VO. Application run-time environment is built from certified resources allocated by standard Grid jobs submitted before or during the experiment.

The IHEP VO is fitted with a portal, which simplifies usage and management of the VO. The portal [5] is used by three user groups: VO managers, site managers and experiment operators. The VO manager adjusts VO by selecting sites participating in the experiment. He also prepares job submission profiles, which describe available application flavors. The VO manager evaluates site's operation according to statistics provided by the portal and confronts them to the signed SLA. The site manager registers site in the VO and can check site's statistics. The experiment operator prepares application's run-time environment by submitting Grid jobs before the experiment starts. During the experiment, he can request additional resources or release already allocated resources by stopping Grid jobs.

3.2. Monitoring Environment

As a monitoring system for the HEP application we chose JIMS, the JMX-based Infrastructure Monitoring System. JIMS is a modern monitoring system equipped with specialized interface following the WS-Management standard (a standard protocol for managing resources through the web) [6]. Built on top of the Java Management Extensions (JMX [7, 8], JIMS is a flexible framework that allows deployment of various modules, like monitoring of Grid infrastructure as well as modules facilitating discovery of monitored nodes within the cluster and the Grid. More, JIMS communication layer is also composed of modules as for example JSR 262 connector — a Web Services connector for JMX technology [6] implementing the aforementioned WS-Management standard.

For our purpose JIMS system was enhanced with P2P capabilities based on Global Discovery Protocol [9]. The Global Discovery Service (GDS) module allows JIMS agents to discover all the agents running within the Grid. Having the list of discovered agents, another module, JIMS Gateway, allows connecting to any other agents from one, arbitrarily chosen agent. In our case this is the agent that is selected and configured in RTD, which collects the monitoring information from one point of attachment with JIMS monitoring system via WS-Management protocol.

For purposes of HEP application, JIMS system was equipped with two types of monitoring modules. First is the Network Monitoring module, monitoring the available bandwidth using the algorithm of measuring dispersion between pairs of packets [10]. The concept of available bandwidth monitoring is used by the senders/receivers components of JIMS agents, communicating with each other in order to measure the available possible bandwidth between HEP application at CERN and the computing resources in clusters all over the Europe (IFCA, CYFRONET, etc.).

The second module allows the RTD to obtain information about computing infrastructure utilization. The OCM-G [13, 14] monitoring system is exploited here. The module, being an OCM-G client, has an access to monitoring information collected by OCM-G's monitors which are started on worker nodes together with HEP application processes. The module interface allows to obtain information about resource usage of these worker nodes e.i. to get the load average values (i.e. the number of jobs in the run queue or waiting for disk I/O averaged over 1, 5, and 15 minutes), an amount of available main memory and an amount of remaining swap space.

3.3. Real-Time Dispatcher

In general, the Grid environment does not support on-line processing. The middleware allows for batch processing, i.e. the user can submit a task, which typically consists of an appropriate program and input data. The Grid middleware queues the task, waiting for available computation resources. Once the Resource Broker assigns a computation node, the task is started and after completing the processing, results are sent back to the user. Therefore, the Real-Time Dispatcher enabling online processing has been developed to support usage of the Grid environment.

This process consists in starting a given number of RemotePT tasks on the Grid and registering them in the RTD. Those tasks are waiting for the data stream to arrive. When the data arrive to the EFD and pointer is passed to ProxyPT, ProxyPT requests a connection to RemotePT from the RTD. It passes the IP address and port on which it is listening. The RTD selects an appropriate RemotePT instance, passes ProxyPT information and a connection between them is established. Thus, it is possible to process data generated in the experiment on the Grid.

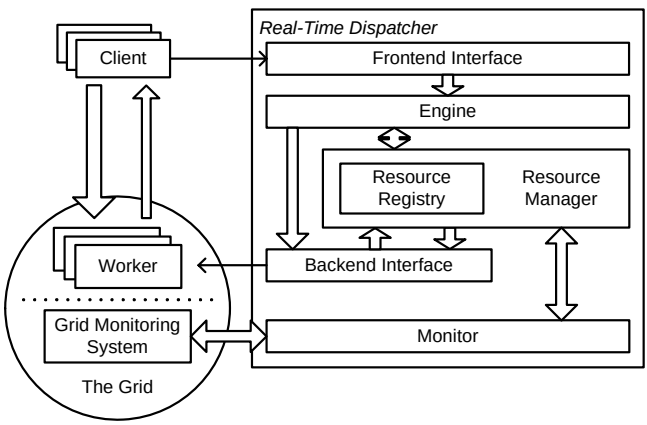

Fig. 2. The Dispatcher architecture

The internal structure of the RTD is shown in Figure 2. It comprises of several modules, each with well defined functionality:

- Frontend – receives requests from ProxyPT,
- Backend – maintains communication with RemotePT,
- Resource Manager – manages available RemotePTs,
- Monitor – collects monitoring data.

The RTD constantly monitors Grid resources where remotePTs instances have been launched and maintains list of nodes with history of measurements. It uses the Grid middleware solutions JIMS and OCM-G to get the data. The metrics currently used are the average load, free memory and free swap memory. When the proxyPT request to find a remotePT arrives to the RTD, the latter uses the set of monitoring data to find the best destination for the request. The RTD replies to the proxyPT with IP address and port number of selected node. These data are then used by the proxyPT to communicate directly with remotePT.

4. Network QoS as feasibility studies on using remote farms for online processing

In preparation for tests of the proposed architecture we measured QoS parameters of the computer network between CERN, Geneva, Switzerland and the remote farm located in Academic Computer Center CYFRONET AGH, Krakow, Poland. These measurements focused on latency, number of lost packets and possible assymetry in both directions. We used system ANT — a setup with two computers located at both ends of the connection equipped with programmable network interfaces and interfaces to the GPS system [11]. The GPS allowed to synchronize machines located thousands kilometers apart and measure parameters in either of the directions separately. To confirm our measurements we also used iperf [12]. The measurements indicated that the network QoS parameters were satisfactory for using CYFRONET remote farm for on-line processing. The latency was constant and the fraction of packet loss was negligible. The latter is very important as even small fraction of lost packets may results in significant limitation is effective transfers and underusage of installed network capacity. Results from packet loss measurements are preseted in Figure 3. The higher loses reported by iperf for the smallest packets have been identified as problems with buffering at the operating system level.

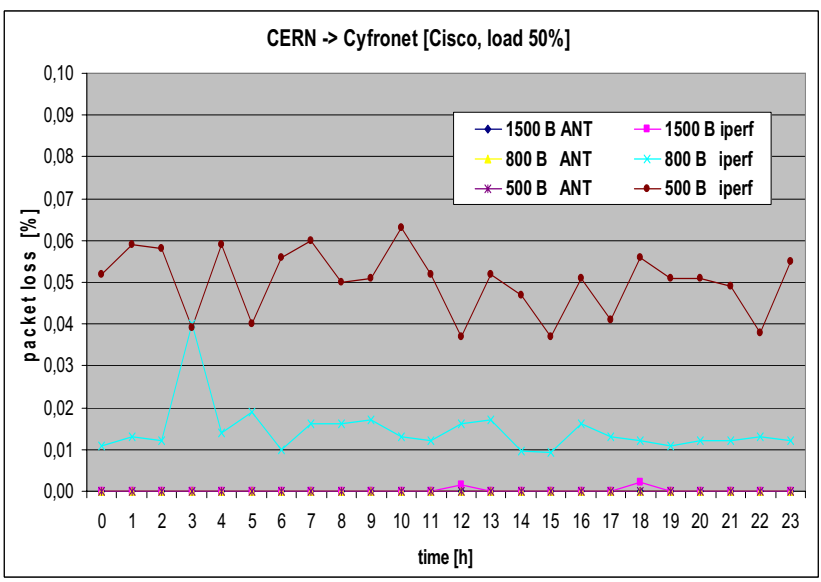

Fig. 3. Fraction of lost packets as function of day time in 24 h run of sending packets from CERN to Cyfronet with 50% network load, various packet sizes. Measurements were taken using two methods: ANT and iperf

5. Experiment results

To evaluate performance of the proposed extension to the ATLAS TDAQ system we run set of measurements using the original TDAQ system and refer to them as baseline performance. We added, to the TDAQ system database, the IP addresses of a number of machines located at CYFRONET and statically assigned to them the EFD and PT tasks keeping the SFI and SFO located in CERN. In our measurements we used 1.6 MB event size, and multithreaded TCP transfers (30 was found to be minimal number of threads for saturated TPC performance). In the Figure 4. we present event processing rate as function of event processing time. For short processing times the saturation originates from the network throughput limitation reaching 70% of the Gigabit nominal value. For longer processing times the event rate enters hyperbole. For processing times expected in ATLAS the system scales linearly; doubling the number of PTs allows for two times longer processing time with the same event rate.

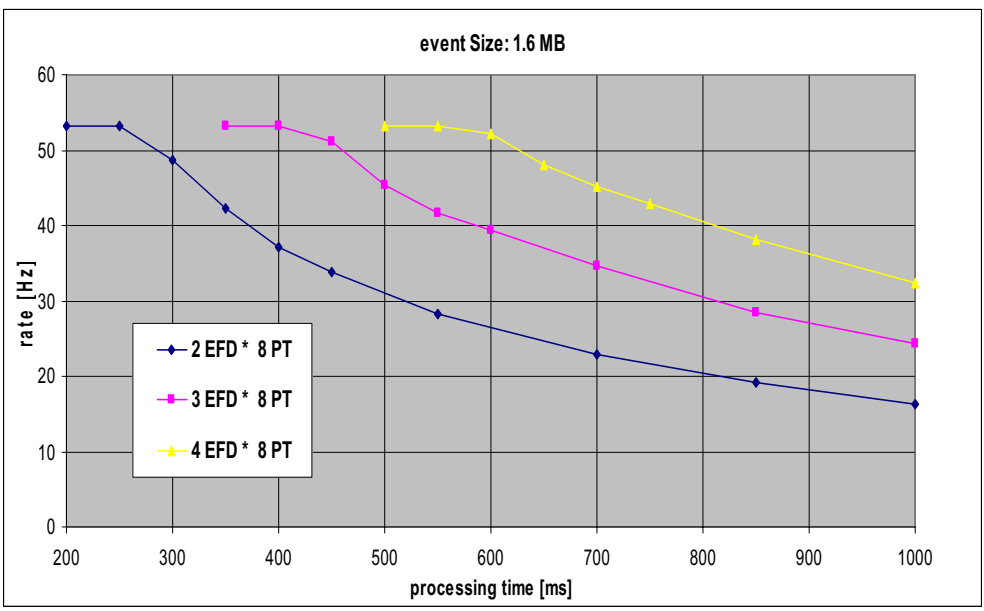

Fig. 4. Event processing rate as function of processing time for the ATLAS baseline setup. For short processing times the processing rate is limited by the network capacity; for longer it is reciprocal to the duration of processing time. For processing times expected in ATLAS, the system scales linearly for a given processing time, doubling number of PTs doubles event processing rate

The results of experiment performed on a system extended by the Real-Time Dispatcher are presented in Figure 5. The RTD is an extension of the ATLAS TDAQ and the overall system effectiveness, in the sense of the frequency, is lower than for the baseline ATLAS, because there are additional data transfers. In baseline system

the data are moved from SFI to EFD where local PTs run analyses software. For the RTD there are additional transfers between proxyPT and remotePT. These additional transfers degrade effectiveness of the RTD for short processing times when the network performance dominates the overall event processing rate. For longer processing times (for the ATLAS it is expected to be in range 1–4 seconds) the system with RTD should outperform the baseline as it should be able to dynamically select nodes with best performance.

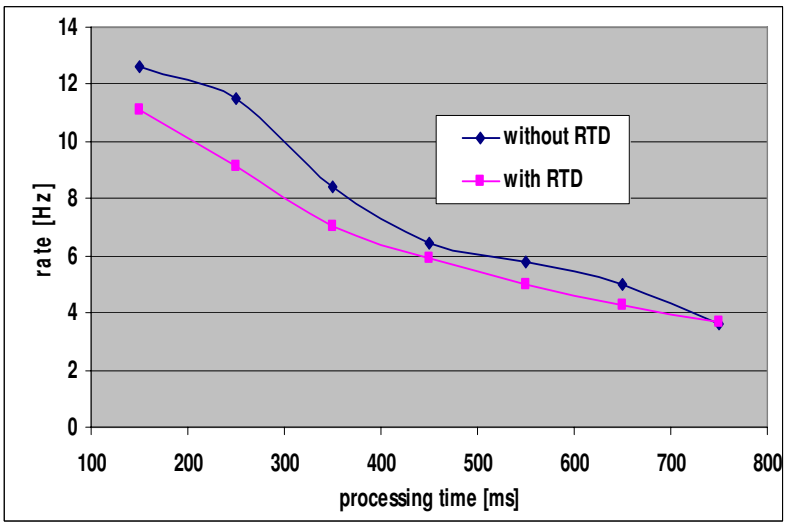

Fig. 5. Comparison of the event processing rate as function of processing time between system without (diamonds) and with (squares) RTD. At the early stage of the RTD development the TCP transmission was using only one thread, hence the comparison measurements were performed with single thread transmissions

6. Conclusions and future work

The presented solution is an extension of the ATLAS TDAQ software that makes possible to use the Grid infrastructure for the on-line processing. Our work focused on identifying the optimal place to break standard ATLAS PT task into the local proxyPT and the remotePT which can be submitted to remote computing resources. We also designed and implemented first version of the Real-Time Dispatcher. We compared the performance against the standard ATLAS TDAQ software with static configuration. For the expected ATLAS processing times, the performance of the two systems is the same. The further development of the RTD should be aimed in improving TCP transmissions and dynamically distribute events based on information collected from the Grid monitoring infrastructure.

Acknowledgements

The work described in this paper was supported in part by the European Union through the IST-2006-031857 project "Interactive European Grid".

References

[1] ATLAS HOME PAGE: `http://atlas.web.cern.ch/Atlas/index.html`

[2] LHC HOME PAGE: `http://lhc.web.cern.ch/lhc`

[3] Skitał Ł., Dutka Ł., Korcyl K., Janusz M., Słota R., Kitowski J.: *Virtual Organization approach for running HEP applications in Grid Environment*. Cracow Grid Workshop, October 15–18, 2006, Cracow, Poland

[4] Skitał Ł., Janusz M., Słota R., Kitowski J.: *Service Level Agreement metrics for real-time application on the Grid*. PPAM 2007, September 9–12, 2007, Gdansk, Poland

[5] Skitał Ł., Słota R., Janusz M., Kitowski J.: *Management of Virtual Organisation for demanding applications in the Grid Environment*. CGW 2007, October 15–17, Kraków, Poland

[6] Denise J-F., Fuchs D.: *Java Management Extensions (JMX) Interoperation With Non Java Technologies*. Available at: `http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/JSR262\_Interop.pdf`, 2007

[7] Sun Microsystems: *Java Management Extensions (JMX) Specification, version 1.4, JSR 160*. Available at: `http://jcp.org/en/jsr/detail?id=160jmx1_4mrel3spec.pdf`, Canta Clara, CA, 2006, retrieved: Dec. 2006

[8] Sun Microsystems: *Java Management Extensions Instrumentation and Agent Specification v. 1.2, JSR 003*. Available at: `http://jcp.org/en/jsr/detail?id=3jmx_1.2_spec.pdf`, Santa Clara, CA, 2002, retrieved: Dec. 2006

[9] Wojtas K., Wasilewski L., Balos K., Zielinski K.: *Discovery Service for JMX-Enabled Monitoring System. JIMS Case Study*. CGW'05 Workshop Proceedings, ACC CYFRONET AGH, Kraków, 2006, pp. 148–157

[10] Dovrolis C. *et al.*: *What Do Packet Dispersion Techniques Measure?* Infocom, 2001, pp. 905–914

[11] Korcyl K., Beuran R., Dobinson B., Ivanovici M., Maia L.M., Meirosu C., Sladowski G.: *Network Performance Measurements as Part of Feasibility Studies on Moving an ATLAS Event Filter to Off-Site Institutes*. LNCS 2970, 2003, pp. 206–213

[12] IPERF project home page. Available at: `http://dast.nlanr.net/projects/Iperf`.

[13] OCM-G project home page. Available at: `http://grid.cyfronet.pl/ocmg/`

[14] Funika W., Guzy K.: *Integration of OCM-G into the JIMS infrastructure for the monitoring of a HEP application*. Proceedings of Cracow Grid Workshop – CGW'07, October 15–17, 2007, Cracow, Poland